AlarmItem Custom Generation Code

Generate alarms dynamically in code.

This document describes how to configure a solution to generate alarms using the Scripts Module, instead of doing the configuration at the AlarmsItems session. The methods presents here allows to Activate, Normalize, and Acknowledge alarm items directly from your customization scripts, bypassing the AlarmsItems configuration table, but still using the AlarmGroup properties, AlarmViewer Online table, and saving to AlarmHistory

Configuration
- Script Methods
- AlarmCustomInfo Settings
- Alarm Initial Condition
- Alarm Acknowledgement

Execution
Example
- In this section...

Configuration

Script Methods

Those methods are on the Alarm.Group namespace. Example: Alarm.Group.Warning.ActivateCustomAlarmsAsync(...

Custom Alarm Methods

```
/// <summary>
/// Activate custom alarms
/// </summary>
/// <param name="items">Array of AlarmCustomInfo with information to Activate alarms</param>
public virtual async Task ActivateCustomAlarmsAsync(AlarmCustomInfo[] items)

/// <summary>
/// Normalize custom alarms
/// </summary>
/// <param name="itemNames">Array of ItemNames to normalize</param>
public virtual async Task NormalizeCustomAlarmsAsync(string[] itemNames)

/// <summary>
/// Acknowledge custom alarms
/// </summary>
/// <param name="itemNames">Array of ItemNames to Acknowledge</param>
public virtual async Task AcknowledgeCustomAlarmsAsync(string[] itemNames)
```

AlarmCustomInfo Class

```
// ID of the item to be activated as an alarm
public string ItemName;

// Name of the tag associated with the alarm
public string TagName;

// Condition of the alarm (integer from eCondition enumeration)
public int Condition;

// Limit of the alarm (for Hi, HiHi, Lo, LoLo conditions)
public string Limit;

// Message to be shown when the alarm is activated
public string Message;

// Alarm area
public string Area;

// Priority of the alarm (0–5 recommended), or any integer value
public int Priority;

// Time when the alarm was activated (UTC time)
public DateTime UtcActiveTime;

// Value of the item when the alarm was activated
public string Value;

// Category of the alarm, from Solution Categories (e.g. "Process", "Safety", "Maintenance", etc.)
public string Category;

// Auxiliary value of the alarm, general use
public string AuxValue;

// Auxiliary value of the alarm, general use
public string AuxValue2;

// Auxiliary value of the alarm, general use
public string AuxValue3;
```

Condition Enumeration

```
   /// <summary>
   /// Alarm Item condition
   /// </summary>
   public enum eCondition { Hi = 0, HiHi = 1, Lo = 2, LoLo = 3, RateOfChange = 4, DeviationMinor = 5,
DeviationMajor = 6, Equal = 7, GreaterThan = 8, GreaterEqual = 9, LessThan = 10, LessEqual = 11, Changed =
12, ChangedUp = 13, ChangedDown = 14, NotEqual = 15 }
```

AlarmCustomInfo Settings

The key parameter of the AlarmCustomInfo is the **ItemName**.

The ItemName is defined by user, it cannot start with "ID", as those are reserved to the alarms created by Alarm Engine, based on the AlarmsItems table.

We recommend to keep similar standard, having two letters that related to the data source, then a sequential number.

ItemName Property

The ItemName must unique to each alarm, as repetitive ItemName will be understood as the same alarm (if already activated, you can't active it again.

The ItemName is the key used in the methods do normalize or acknowledge the tag from script.

Except by the ItemName and Condition, all the other fields are optional, but we strongly recommend to use then consistency with the configuration of the alarms in the AlarmItems table, and the columns you intend to visualize in the AlarmViewer graphical component.

Alarm Initial Condition

If the alarm is created with Condition *Changed, ChangedUp, or ChangedDown*, as they represent an EVENT, not a STATE, they are generated with its status in NORMALIZED, meaning they only stay on the alarm online page if the group required acknowledgement, otherwise they go directly to the AlarmHistory.

The alarms created with Condition Hi, HiHi, Lo, .. and all the conditions that reflect an STATE, are generated with the status ACTIVE. They will stay in the alarm online page until you call the script method to normalize them (using the ItemName to define which alarm to normalize).

Alarm Acknowledgement

When the alarm is presented in the AlarmViewer graphical components, it can be acknowledged directly from the commands available on that User Interface. Otherwise you can use teh custom acknowledge method to do it directly .

The alarms stay in the AlarmOnline view when they have Acknowledment pending, even if they are normalized, if the AlarmGroup requires ack.

The alarms that in Normalized state, and don't require ACK (or have been acknowledged) will present only on the AlarmHistorian (assuming the AlarmGroup is set to log those events in the AlarmHistorian database).

Execution

The alarms show in the AlarmViewer online, and are stored to the AlarmHistorian, in the same way the alarms created on AlarmItems would behave.

There are though some restrictions on the use of those custom alarms:

- In hot-standby scenarios the Alarm Online not updated with the custom alarms, on the switch. A custom script is necessary to enable it.
- The internal counters Alarm.TotalCount and other counters, as the Tag Alarm properties, will not update base on those messages.

Essentially, the custom alarm are intended to be used only in local Online Alarms table, and Acknowledgement and Logging if the group is enabled to those actions.

Example

Solution Example

Custom Alarms Example

In this section...
